МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ компьютерной безопасности и криптографии

**Проверка чисел на простоту**

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ТЕОРЕТИКО-ЧИСЛОВЫЕ МЕТОДЫ В КРИПТОГАФИИ»

студента 5 курса 531 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Серебрякова Алексея Владимировича

|  |  |  |
| --- | --- | --- |
| Преподаватель  профессор, д.ф.-м.н. | \_\_\_\_\_\_\_\_\_\_ | В. А. Молчанов |
|  | подпись, дата |  |

Саратов 2023

Оглавление

[Введение 3](#_Toc151306293)

[Цель работы и порядок её выполнения 4](#_Toc151306294)

[1 Теоретическая часть 5](#_Toc151306295)

[1.1 Тест Ферма 5](#_Toc151306296)

[1.2 Tест Соловея-Штрассена 6](#_Toc151306297)

[1.3 Тест Миллера-Рабина 7](#_Toc151306298)

[2 Псевдокоды программ 9](#_Toc151306299)

[2.1 Псевдокод алгоритма проверки на простоту по тесту Ферма 9](#_Toc151306300)

[2.2 Псевдокод проверки на простоту по тесту Соловея-Штрассена 9](#_Toc151306301)

[2.3 Псевдокод проверки на простоту по тесту Миллера-Рабина 9](#_Toc151306302)

[3 Тестирование программы 10](#_Toc151306303)

[ПРИЛОЖЕНИЕ А 13](#_Toc151306304)

Введение

В данной лабораторной работе поставлена задача рассмотрения теста Ферма, Соловея-Штрассена, Миллера-Рабина, написание алгоритмов для изученных тем.

Цель работы и порядок её выполнения

Цель работы – изучение основных методов проверки простоты чисел и их программная реализация.

Порядок выполнения работы:

1. Рассмотреть тест Ферма проверки чисел на простоту и привести его программную реализацию.
2. Рассмотреть тест Соловея-Штрассена проверки чисел на простоту и привести его программную реализацию.
3. Рассмотреть тест Миллера-Рабина и привести его программную реализацию.

1 Теоретическая часть

1.1 Тест Ферма

Малая теорема Ферма. Если – простое число, то для любого выполняется свойство .

Алгоритм 1 – тест простоты на основе малой теоремы Ферма:

Вход. Нечетное число .

Выход. “Число вероятно, простое” или “Число составное”.

Шаг 1. Выбрать случайно и вычислить . Если , то ответ “Число составное”.

Шаг 2. Если , то проверить условие . Если оно не выполнено, то ответ “Число составное”. В противном случае ответ “Число , вероятно, простое”.

Трудоемкость алгоритма: .

Определение. Число называется псевдопростым по основанию , если выполняется . Здесь .

Лемма 1. Для нечетного числа справедливы утверждения:

– подгруппа ;

если , то по теореме Лагранжа .

Вероятность успеха – вероятность получить “Число составное” для составного числа равна . Возможны три случая:

1. число простое и тест всегда дает ответ “Число , вероятно, простое”;
2. число составное и , тогда тест дает ответ “Число составное” с вероятностью успеха ;
3. число составное и , тогда тест дает ответ “Число составное” с вероятностью успеха .

В случае 2 при повторах теста вероятность успеха .

1.2 Tест Соловея-Штрассена

Критерий Эйлера. Нечетное число является простым тогда и только тогда, когда для любого выполняется свойство:

Причем – простое число где .

Алгоритм 2 – тест простоты Соловея-Штрассена

Вход. Нечетное число .

Выход. “Число , вероятно, простое” или “Число составное”.

Шаг 1. Выбрать и вычислить . Если , то ответ “Число составное”.

Шаг 2. Если , то проверить условие . Если оно не выполнено, то ответ “Число составное”. В противном случае ответ “Число , вероятно, простое”.

Трудоемкость алгоритма: .

Определение. Число называется эйлеровым псевдопростым по основанию , если выполняется .

Лемма 1. Для нечетного числа справедливы утверждения:

* – подгруппа ;
* Если – составное число, то .

Вероятность успеха Алгоритма 2 для составного числа равна Возможны два случая:

1. число простое и тест всегда дает ответ “Число , вероятно, простое”;
2. число составное и тест дает ответ “Число составное” с вероятностью успеха .

В случае 2 при повторах теста вероятность успеха

1.3 Тест Миллера-Рабина

Теорема (Критерий Миллера). Пусть – нечетное число и для нечетного . Тогда является простым в том и только том случае, если для любого выполняется свойство

Причем – простое число где .

Необходимость: для простого выполняется:

Алгоритм 3 – тест простоты Миллера-Рабина

Вход. Нечетное число .

Выход. “Число , вероятно, простое” или “Число составное”.

Шаг 1. Выбрать и вычислить . Если , то ответ “Число составное”.

Шаг 2. Если , то вычислить для значений . Если или для некоторого , то ответ “Число , вероятно, простое”. В противном случае ответ “Число составное”.

Трудоемкость алгоритма: .

Определение. Число , псевдопростое по основанию , называется сильно псевдопростым по этому основанию , если выполняется , т.е. выполняется одно из условий:

;

для некоторого .

Для составного числа выполняется и, значит, вероятность успеха Алгоритма 3 для составного числа равна . При повторах теста вероятность успеха:

2 Псевдокоды программ

2.1 Псевдокод алгоритма проверки на простоту по тесту Ферма

Процедура проверкичисла:

:

Вернуть “Число составное”

Установить как случайное значение в

Вычислить

:

Вернуть “Число составное”

:

Проверить условие

условие не выполнено:

Вернуть “Число составное”

Иначе:

Вернуть “Число , вероятно, простое”

2.2 Псевдокод проверки на простоту по тесту Соловея-Штрассена

Процедура проверкичисла:

допустим случайное число из

допустим

:

Вернуть “Число составное”

:

условие не выполнено:

Вернуть “Число составное”

Иначе:

Вернуть “Число , вероятно, простое”

2.3 Псевдокод проверки на простоту по тесту Миллера-Рабина

Процедура проверкичисла:

Убедиться, что > и нечетное.

Выбрать из

Вычислить

:

Вернуть “Число составное”

:

Вычислить и , такие что и нечетное.

Для каждого в :

Вычислить

или

Вернуть “Число , вероятно, простое”

Вернуть “Число составное”

3 Тестирование программы

На рисунках 1-3 представлены результаты работы программы при проверке некоторых чисел на простоту. На рисунке 4 представлен случай, когда разные тесты выдают разный результат при проверке одного и того же числа. Это происходит потому, что тесты являются вероятностными и для достижения единого ответа потребуется большее количество проверок.

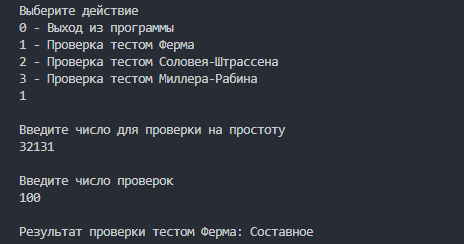


Рисунок 1 - Проверка на простоту тестом Ферма

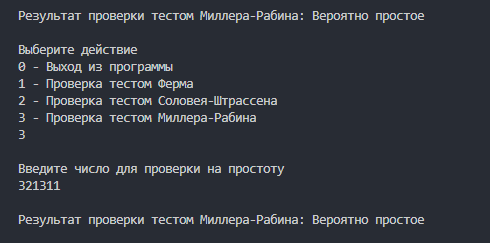


Рисунок 2 - Проверка на простоту тестом Соловея-Штрассена

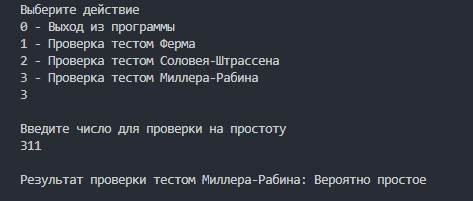


Рисунок 3 - Проверка на простоту тестом Миллера-Рабина

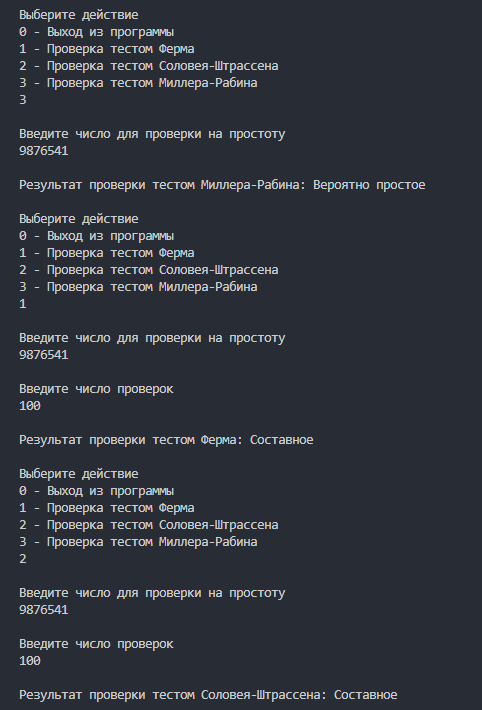


Рисунок 4 - Проверка одного числа на простоту тестами Ферма, Соловея-Штрассена и Миллера-Рабина

ПРИЛОЖЕНИЕ А

**Листинг программы lab3.rb**

require "time"

def timer(f)

lambda do |\*args, \*\*kwargs|

start = Process.clock\_gettime(Process::CLOCK\_MONOTONIC)

ans = f.call(\*args, \*\*kwargs)

finish = Process.clock\_gettime(Process::CLOCK\_MONOTONIC)

puts "Time: #{(finish - start) \* 1000}ms"

ans

end

end

def hlp\_ferma(a, n)

ans = 1

(1..n-1).each do |i|

ans = (ans \* a) % n

end

ans = (ans - 1) % n

ans

end

def fermat(n, k)

k.times do

a = rand(n - 2) + 2

return 0 if a.gcd(n) > 1

tmp = hlp\_ferma(a, n)

return 0 if tmp != 0

end

1

end

def hlp(base, e, n)

x, y = 1, base

while e != 0

if e % 2 != 0

x = (x \* y) % n

end

y = (y \* y) % n

e = e / 2

end

x % n

end

def jacobi(a, n)

return 0 if a.gcd(n) != 1

t = 1

a %= n

while a != 0

while a % 2 == 0

a /= 2

if n % 8 == 3 || n % 8 == 5

t = -t

end

end

n, a = a, n

if a % 4 == 3 && n % 4 == 3

t = -t

end

a %= n

end

t

end

def solovay\_strassen(n, k)

return false if n < 2

return false if n != 2 && n % 2 == 0

k.times do

a = rand(n - 1) + 1

jacobian = (n + jacobi(a, n)) % n

mod = hlp(a, (n - 1) / 2, n)

if jacobian == 0 || mod != jacobian

return false

end

end

true

end

def miller\_rabin(n)

return 1 if n == 2 || n == 3

return 0 if n.even? && n != 2

s, t, x = 0, n - 1, 0

r1, r2 = 2, n - 2

a, r = Math.log2(n).to\_i, Math.log2(n).to\_i

while t != 0 && t % 2 == 0

s += 1

t /= 2

end

r.times do

a = r1 + rand(r2 - r1)

x, j = 1, 1

while j <= t

x = (x \* a) % n

j += 1

end

next if x == 1 || x == n - 1

(s - 1).times do

x = (x \* x) % n

return 0 if x == 1

break if x == n - 1

end

return 0 if x != n - 1

end

1

end

def main

opt = 1

while opt != 0

puts "\nВыберите действие"

puts "0 - Выход из программы"

puts "1 - Проверка тестом Ферма"

puts "2 - Проверка тестом Соловея-Штрассена"

puts "3 - Проверка тестом Миллера-Рабина"

opt = gets.strip.to\_i

break if opt == 0

puts "\nВведите число для проверки на простоту"

n = gets.strip.to\_i

if opt == 1 || opt == 2

puts "\nВведите число проверок"

k = gets.strip.to\_i

end

case opt

when 1

r = fermat(n, k)

if r == 0

puts "\nРезультат проверки тестом Ферма: Составное"

else

puts "\nРезультат проверки тестом Ферма: Вероятно простое"

end

when 2

r = solovay\_strassen(n, k)

if !r

puts "\nРезультат проверки тестом Соловея-Штрассена: Составное"

else

puts "\nРезультат проверки тестом Соловея-Штрассена: Вероятно простое"

end

when 3

r = miller\_rabin(n)

if !r

puts "\nРезультат проверки тестом Миллера-Рабина: Составное"

else

puts "\nРезультат проверки тестом Миллера-Рабина: Вероятно простое"

end

end

end

end

if \_\_FILE\_\_ == $PROGRAM\_NAME

main

end